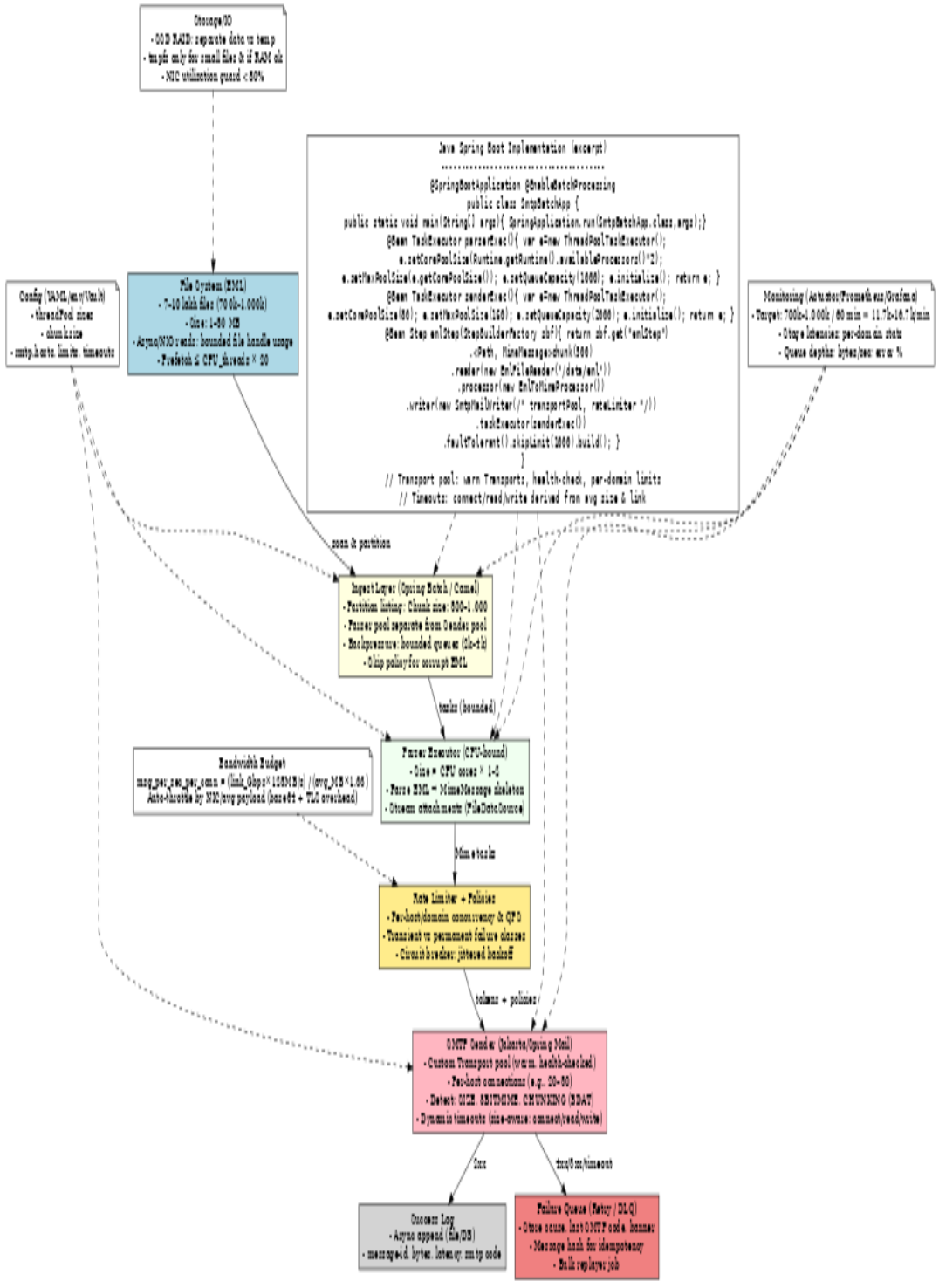
SMTP Design with Java Implementation



Page 1 shows the high-level design diagram for sending 7–10 lakh EML files in 1 hour using a Spring Boot SMTP pipeline. The next page provides the Java Spring Boot implementation snippet.

Java Spring Boot Implementation Snippet

```
@SpringBootApplication
@EnableBatchProcessing
public class SmtpBatchApp {

    public static void main(String[] args) {
        SpringApplication.run(SmtpBatchApp.class, args);
    }

    @Bean
    TaskExecutor parserExec() {
        ThreadPoolTaskExecutor e = new ThreadPoolTaskExecutor();
        e.setCorePoolSize(Runtime.getRuntime().availableProcessors() * 2);
        e.setMaxPoolSize(e.getCorePoolSize());
        e.setQueueCapacity(1000);
        e.initialize();
        return e;
    }

    @Bean
    TaskExecutor senderExec() {
        ThreadPoolTaskExecutor e = new ThreadPoolTaskExecutor();
        e.setCorePoolSize(80);
        e.setMaxPoolSize(160);
        e.setQueueCapacity(2000);
        e.initialize();
        return e;
    }

    @Bean
    Step emlStep(StepBuilderFactory sbf) {
        return sbf.get("emlStep")
            .<Path, MimeMessage>chunk(500)
            .reader(new EmlFileReader("/data/eml"))
            .processor(new EmlToMimeProcessor())
            .writer(new SmtpMailWriter(/* transportPool, rateLimiter */))
            .taskExecutor(senderExec())
            .faultTolerant().skipLimit(1000)
            .build();
    }
}

// Notes:
// - Transport pool: warm Transports, health-check, per-domain limits
// - Timeouts: connect/read/write derived from avg size & link
```